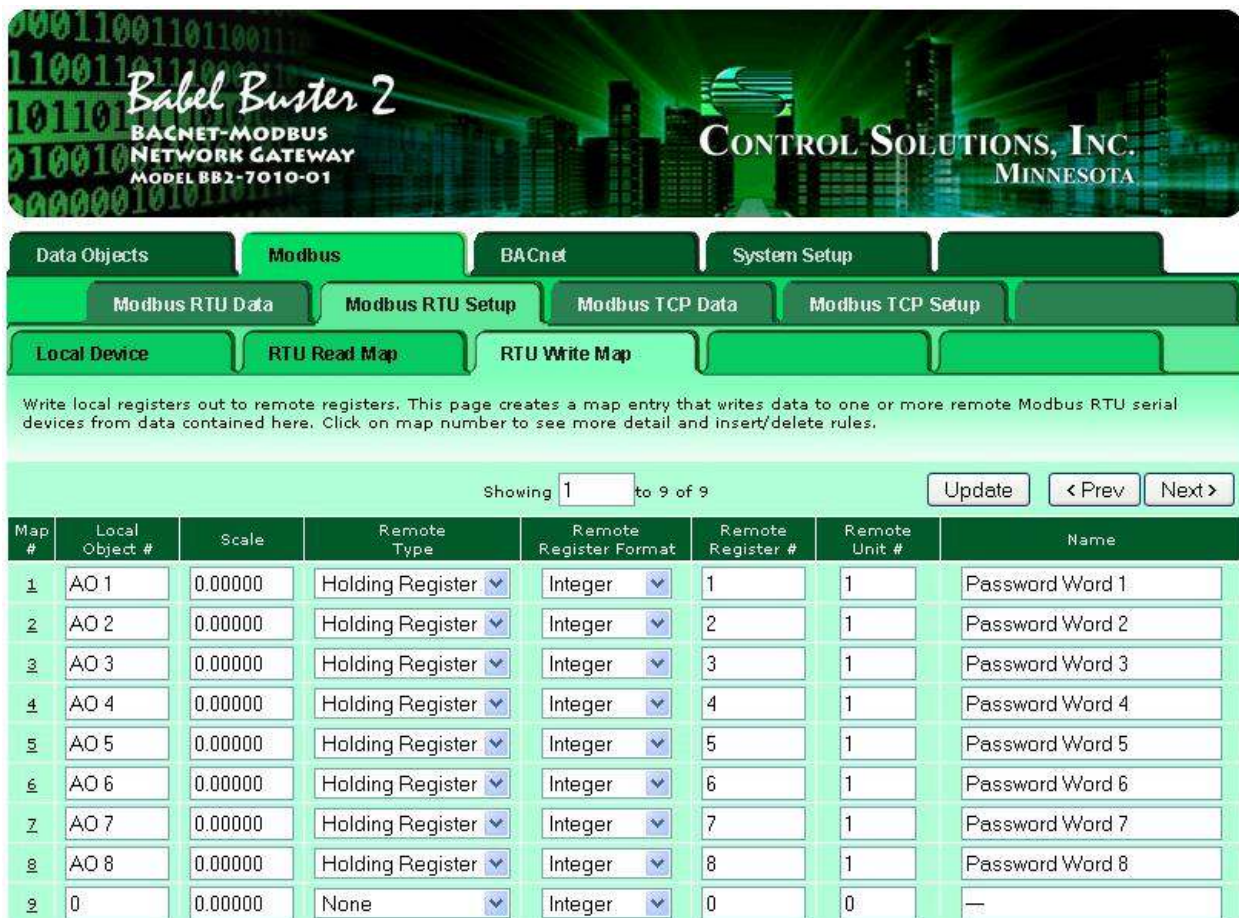


# How to write a Modbus password from BACnet using a Babel Buster BB2-7010

## How to write a Modbus password from BACnet using a Babel Buster BB2-7010

There is sometimes a requirement to write a ‘password’ or some sort of unlock code to a Modbus device. This unlocking usually requires writing a series of registers in one single request, generally with function code 16. The BB2-7010 includes the options necessary to accomplish this.

First, determine the number of registers you will need to write. If the Modbus device talks about a "16-character" password, that means 8 registers with 2 characters per register. In the example that follows, we are assuming a 16-character password. This means we need to define 8 consecutive write maps. We will illustrate RTU, but the same principle applies to TCP.



The screenshot shows the Babel Buster 2 software interface. The title bar reads "Babel Buster 2 BACNET-MODBUS NETWORK GATEWAY MODEL BB2-7010-01" and "CONTROL SOLUTIONS, INC. MINNESOTA". The navigation menu includes "Data Objects", "Modbus", "BACnet", and "System Setup". Under "Modbus", there are sub-menus for "Modbus RTU Data", "Modbus RTU Setup", "Modbus TCP Data", and "Modbus TCP Setup". The "Modbus RTU Setup" menu is expanded to show "Local Device", "RTU Read Map", and "RTU Write Map". The "RTU Write Map" page is active, displaying a table of write maps. The table has columns for Map #, Local Object #, Scale, Remote Type, Remote Register Format, Remote Register #, Remote Unit #, and Name. The table shows 9 rows, with the first 8 rows representing password words and the 9th row representing a blank register.

Write local registers out to remote registers. This page creates a map entry that writes data to one or more remote Modbus RTU serial devices from data contained here. Click on map number to see more detail and insert/delete rules.

Showing 1 to 9 of 9 Update < Prev Next >

Map #	Local Object #	Scale	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Name
1	AO 1	0.00000	Holding Register	Integer	1	1	Password Word 1
2	AO 2	0.00000	Holding Register	Integer	2	1	Password Word 2
3	AO 3	0.00000	Holding Register	Integer	3	1	Password Word 3
4	AO 4	0.00000	Holding Register	Integer	4	1	Password Word 4
5	AO 5	0.00000	Holding Register	Integer	5	1	Password Word 5
6	AO 6	0.00000	Holding Register	Integer	6	1	Password Word 6
7	AO 7	0.00000	Holding Register	Integer	7	1	Password Word 7
8	AO 8	0.00000	Holding Register	Integer	8	1	Password Word 8
9	0	0.00000	None	Integer	0	0	—

You can enter almost everything you will need from the map list page illustrated above. Once you have created your series of registers, proceed to modify as illustrated in the following screen shot.

Note: It is important that consecutive registers be defined in consecutive write maps, and that the next write map following the password is NOT in consecutive order, otherwise more than the password will be sent in that single request.

The BB2-7010 (and all Control Solutions gateways) will attempt to send multiple registers in a single write request when they are found to be consecutive in the list of write maps. For purposes of multiple registers per write request, 'consecutive' means contiguous in the list of write maps, but also means the same device or slave address, and consecutively incrementing register numbers.

The goal is to get a series of 8 registers written in a single request, but only once, and only upon update of a BACnet object. Getting the BB2-7010 to write only once, and only on demand, requires setting each of the 8 write maps as illustrated above. Pay attention to the check boxes – only the box illustrated as checked (ticked) should be checked, and all others should be left off. Also be sure to select the correct radio button after "Repeat this process". It should say "no more than every 0.0 seconds".

The combination of "changed by 0.0" and "no more than every 0.0 seconds" is a special case that tells the BB2-7010 to only write to Modbus when an update is received via BACnet, and write to Modbus regardless of whether BACnet actually changed any values.

The object types used for passwords can be either Analog Values or Analog Output objects. However, there is a particular advantage to using the Analog Output. If you set the configured relinquish default value to contain the password to be written, then you only need to write to the first object in the series to cause the entire series to be sent out via Modbus. Furthermore, because the Modbus password is contained within the relinquish default values, you can get away with only writing Null (or relinquish) to the first output object in the series. The BACnet device doing the update does not even need to know the Modbus password. (If you cannot get your BACnet client to write Null or relinquish on demand, then simply write the actual password value to the first output object. The rest of the password will be filled in by the relinquish default values in the rest of the output objects, assuming they have not been written by anybody and are still in the relinquish state.)

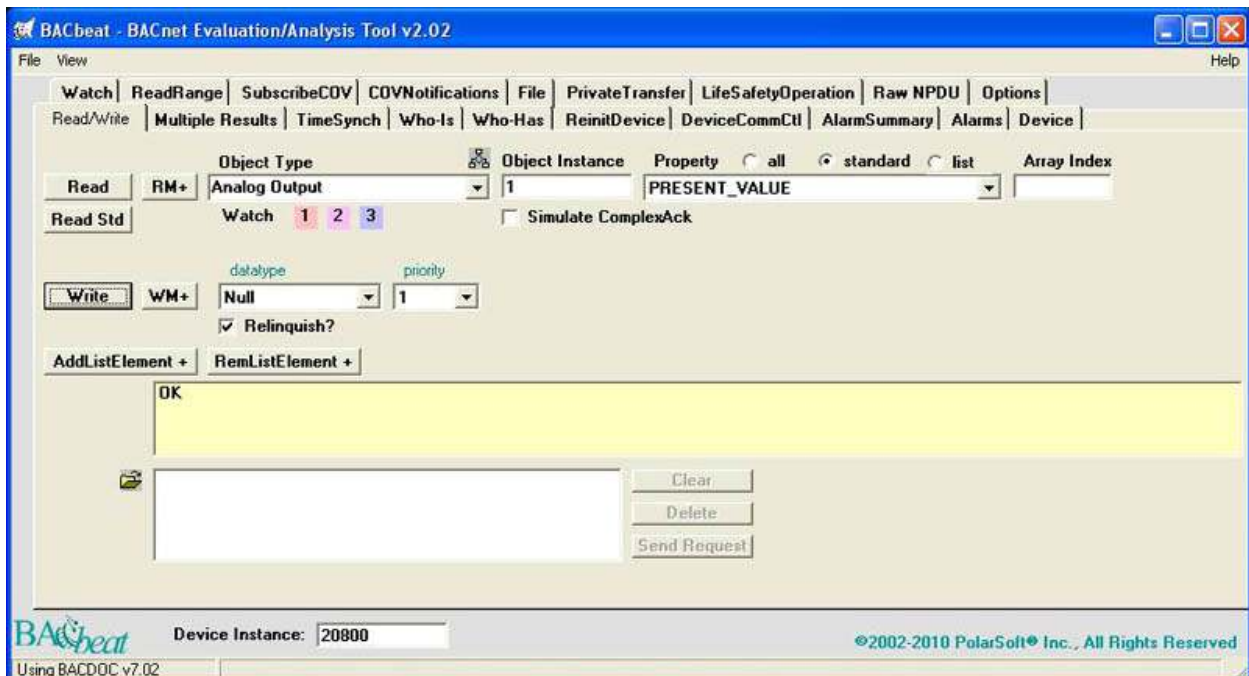
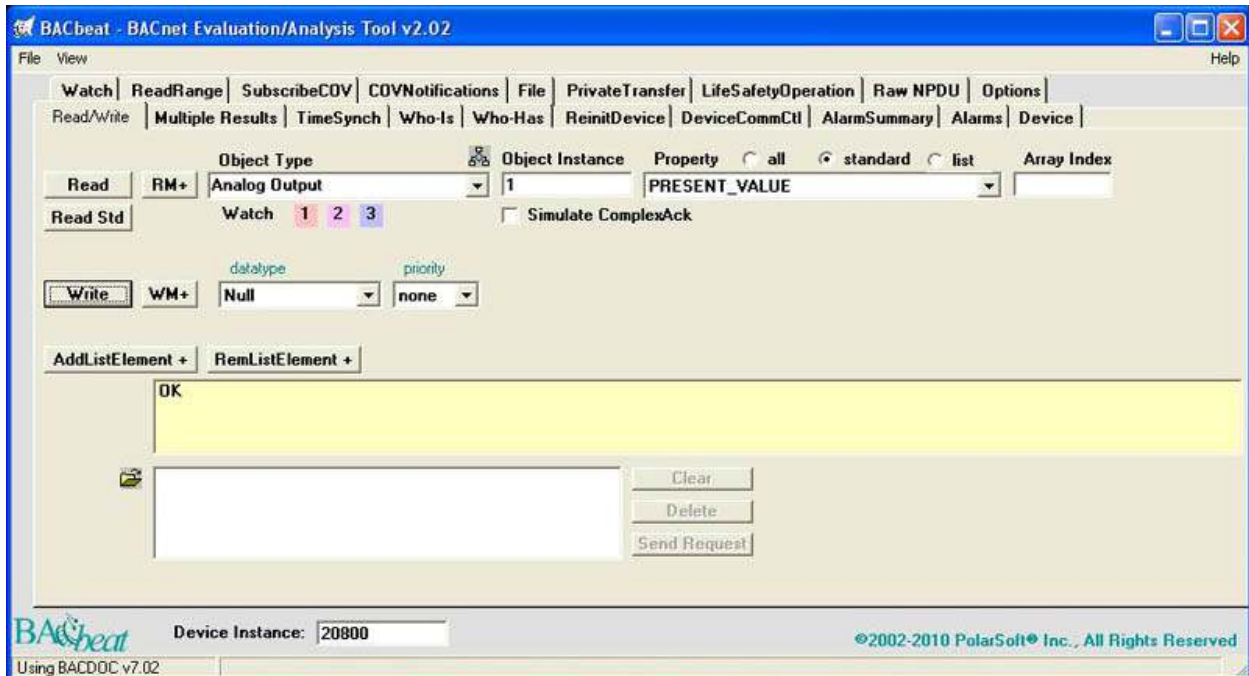
The screenshot shows the 'Babel Buster 2' web interface for a BACNET-MODBUS NETWORK GATEWAY (MODEL BB2-7010-01) by CONTROL SOLUTIONS, INC. MINNESOTA. The interface includes navigation tabs for Data Objects, Modbus, BACnet, and System Setup. Under Data Objects, there are sub-tabs for Analog, Binary, and Multi-State. Further sub-tabs include Input Objects, Output Objects, and Value Objects. A message states: 'This page displays data as presently found in the local objects maintained by this device.' The main content area is titled 'Analog Output Objects' and shows a table of 15 objects. A 'Showing objects from' dropdown is set to '1', with 'Update', '< Prev', and 'Next >' buttons. The table columns are Object, Object Name / Object Description, Out of Service, Present Value, Reliability, Status, and Units.

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
<a href="#">1</a>	Password Word 1	N	16706.0	0	0,0,0,0	no_units
<a href="#">2</a>	Password Word 2	N	17720.0	0	0,0,0,0	no_units
<a href="#">3</a>	Password Word 3	N	17734.0	0	0,0,0,0	no_units
<a href="#">4</a>	Password Word 4	N	18248.0	0	0,0,0,0	no_units
<a href="#">5</a>	Password Word 5	N	16762.0	0	0,0,0,0	no_units
<a href="#">6</a>	Password Word 6	N	19276.0	0	0,0,0,0	no_units
<a href="#">7</a>	Password Word 7	N	19790.0	0	0,0,0,0	no_units
<a href="#">8</a>	Password Word 8	N	20304.0	0	0,0,0,0	no_units
<a href="#">9</a>	Analog Output 9	N	0.00000	0	0,0,0,0	no_units
<a href="#">10</a>	Analog Output 10	N	0.00000	0	0,0,0,0	no_units
<a href="#">11</a>	Analog Output 11	N	0.00000	0	0,0,0,0	no_units
<a href="#">12</a>	Analog Output 12	N	0.00000	0	0,0,0,0	no_units
<a href="#">13</a>	Analog Output 13	N	0.00000	0	0,0,0,0	no_units
<a href="#">14</a>	Analog Output 14	N	0.00000	0	0,0,0,0	no_units
<a href="#">15</a>	Analog Output 15	N	0.00000	0	0,0,0,0	no_units

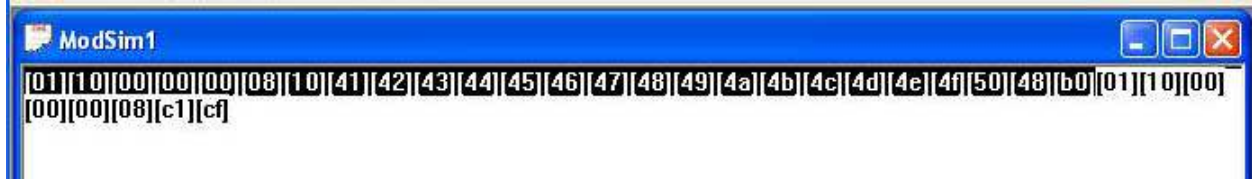
The screen shot above shows a 16-character password set to be written to Modbus. The relinquish default values in the 8 consecutive output objects form ‘ABCDEFGHJKLMNPO’. Click on the object number in the first column to gain access to setting the relinquish default value as illustrated below. When all of the initial relinquish defaults have been entered, go to the Config File page and click ‘Save’ to retain those values. (You will also need to go to the Config File page and click ‘Save’ to retain all of your write map definitions.)



The following screen shots illustrate writing a null, or relinquish, to Analog Output 1.



The result of writing the Null to Analog Output 1, with the write maps defined as first illustrated above, will be writing a series of 8 registers to Modbus using function code 16, illustrated in the following traffic capture on the RTU network:



The bytes 41, 42, 43, etc, are the hexadecimal values for A, B, C, etc. Since two characters will be sent per single Modbus register, you must calculate a value that corresponds to two ASCII characters (assuming the password has been defined as an ASCII string – otherwise use whatever code you are instructed to use by the Modbus device manufacturer).

The letters AB are hexadecimal values 41 and 42. The first character will be in the high order byte. This means the concatenated value will be hex 4142 (or represented often as 0x4142 or 4142H). Now convert this to decimal using a hex to decimal calculator (or use your PC's calculator, enter in hex, and switch to decimal). The decimal number is 16706.

You can also calculate character codes in decimal. The letter A is decimal 65, and B is 66. To calculate the 16-bit value to write via the Modbus register, you would multiply the first character times 256, then add the second character as follows:

$$(65 * 256) + 66 = 16706$$

You can find ASCII code charts by simply doing a web search for "ASCII", or go to <http://en.wikipedia.org/wiki/ASCII> where you will find information including the following chart:

Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	sp	100 0000	100	64	40	@	110 0000	140	96	60	`
010 0001	041	33	21	!	100 0001	101	65	41	A	110 0001	141	97	61	a
010 0010	042	34	22	"	100 0010	102	66	42	B	110 0010	142	98	62	b
010 0011	043	35	23	#	100 0011	103	67	43	C	110 0011	143	99	63	c
010 0100	044	36	24	\$	100 0100	104	68	44	D	110 0100	144	100	64	d
010 0101	045	37	25	%	100 0101	105	69	45	E	110 0101	145	101	65	e
010 0110	046	38	26	&	100 0110	106	70	46	F	110 0110	146	102	66	f
010 0111	047	39	27	'	100 0111	107	71	47	G	110 0111	147	103	67	g
010 1000	050	40	28	(	100 1000	110	72	48	H	110 1000	150	104	68	h
010 1001	051	41	29	)	100 1001	111	73	49	I	110 1001	151	105	69	i
010 1010	052	42	2A	*	100 1010	112	74	4A	J	110 1010	152	106	6A	j
010 1011	053	43	2B	+	100 1011	113	75	4B	K	110 1011	153	107	6B	k
010 1100	054	44	2C	,	100 1100	114	76	4C	L	110 1100	154	108	6C	l
010 1101	055	45	2D	-	100 1101	115	77	4D	M	110 1101	155	109	6D	m
010 1110	056	46	2E	.	100 1110	116	78	4E	N	110 1110	156	110	6E	n
010 1111	057	47	2F	/	100 1111	117	79	4F	O	110 1111	157	111	6F	o
011 0000	060	48	30	0	101 0000	120	80	50	P	111 0000	160	112	70	p
011 0001	061	49	31	1	101 0001	121	81	51	Q	111 0001	161	113	71	q
011 0010	062	50	32	2	101 0010	122	82	52	R	111 0010	162	114	72	r
011 0011	063	51	33	3	101 0011	123	83	53	S	111 0011	163	115	73	s
011 0100	064	52	34	4	101 0100	124	84	54	T	111 0100	164	116	74	t
011 0101	065	53	35	5	101 0101	125	85	55	U	111 0101	165	117	75	u
011 0110	066	54	36	6	101 0110	126	86	56	V	111 0110	166	118	76	v
011 0111	067	55	37	7	101 0111	127	87	57	W	111 0111	167	119	77	w
011 1000	070	56	38	8	101 1000	130	88	58	X	111 1000	170	120	78	x
011 1001	071	57	39	9	101 1001	131	89	59	Y	111 1001	171	121	79	y
011 1010	072	58	3A	:	101 1010	132	90	5A	Z	111 1010	172	122	7A	z
011 1011	073	59	3B	;	101 1011	133	91	5B	[	111 1011	173	123	7B	{
011 1100	074	60	3C	<	101 1100	134	92	5C	\	111 1100	174	124	7C	
011 1101	075	61	3D	=	101 1101	135	93	5D	]	111 1101	175	125	7D	}
011 1110	076	62	3E	>	101 1110	136	94	5E	^	111 1110	176	126	7E	~
011 1111	077	63	3F	?	101 1111	137	95	5F	_					

---

Article ID: 2

Created On: Thu, Dec 6, 2012 at 11:24 PM

Last Updated On: Wed, May 20, 2015 at 2:05 PM